**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INGENIERÍA**
**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

# LABORATORIO 3: PTHREAD

# SISTEMAS OPERATIVOS

**ANDRÉS MUÑOZ**

**SHALINI RAMCHANDANI**

Profesores:

Fernando Rannou

Miguel Cárcamo

Ayudantes:

Nestor Mora

Marcela Rivera

Santiago - Chile

9 de noviembre de 2017

# CAPÍTULO 1. ESTRATEGIA DE PARALELIZACIÓN

En esta parte del informe, se explica como se ideó la solución. Para ello el primer paso es definir las estructuras en las cuales se generan todas las operaciones.

## 1.1 ESTRUCTURAS

Para el programa se crearon tres estructuras diferentes entre ellas:

- Position: Representa un posición dentro de una matriz. Tiene dos números, uno para la fila y otro para la columna.
- Wave: Representa la ola. Dentro de esta estructura se encuentra un arreglo tridimensional de floats, la primera, segunda y tercera dimensión corresponden a número de pasos, cantidad de filas y cantidad de columnas respectivamente. Además, se encuentra un arreglo de barreras, esto sirve para esperar a las hebras en cada paso.
- Thread: Representa a una hebra. Dentro de esta estructura se encuentra un arreglo de posiciones, los cuales sirven para saber cuales son las posiciones que calculará la hebra. Además, se encuentra el tipo de dato pthread, el cual es la hebra en sí.

## 1.2 ESTRATEGIA

La estrategia de paralelización, se centra en que cada hebra siempre calcula las mismas posiciones para cada paso diferente. Para ello es necesrio primero realizar los siguientes pasos:

1. Crear la estructura Wave, especificando el número de pasos, filas, columnas y hebras. Aquí se debe pedir memoria para el arreglo tridimensional de floats y arreglo de barreras. Cabe recalcar que esto requiere mucha memoria ya que se crea una matriz nueva por cada paso que se realiza.
2. Inicializar un arreglo de de estructuras de thread, esto quiere decir que se pide memoria para cierta cantidad de esta estructura y además, se les asigna las diferentes

posiciones en las cuales trabajaran estas. La forma de asignar las posiciones es ir recorriendo la matriz e ir asignando a cada thread una posición cuando ya se les asigna a todos una posición se asigna la segunda posición a todos los thread,se repite esta operación hasta asignar todas las posiciones existentes.

Una vez asignada las posiciones a trabajar para cada hebra, se inicializan estas para que, por cada instancia o paso que se tenga que calcular la matriz, las hebras trabajen sobre las posiciones que les corresponde y calculan los valores según la ecuación de Schrödinger. Por cada instancia que exista, al final de esta se encuentra la barrera para que se esperen al término de la ejecución de todas las hebras y así poder pasar a la siguiente instancia de la matriz, ya que la ecuación al ser recursiva. necesita obligatoriamente los 2 pasos previos de la matriz.

La barrera termina la espera apenas lleguen la cantidad total de hebras, y deja continuar a las hebras para el cálculo de sus posiciones correspondientes a la siguiente instancia, donde a su vez, al igual que en la instancia anterior, se encuentra otra barrera. Esto se repite para todos los pasos a ejecutar.

La implementación de las barreras permite a que la instancia previa de la matriz se llene completa antes de que sea utilizada por la instancia posterior, hecho que si no se cumple podría generar una inconsistencia de datos.

Una vez ejecutado todos los pasos solicitados, se terminan la ejecución de todas las hebras.

# CAPÍTULO 2. RENDIMIENTO COMPUTACIONAL

A continuación, se presentan los gráficos de las hebras con respecto a los tiempos de ejecución, la eficiencia y el Speed Up. Se hace el estudio con las matrices de tamaño 128x128 y 256x256 y con 2000, 4000 y 8000 pasos.

## 2.1. Especificaciones de los Experimentos

Cada prueba fue realizada 5 veces, utilizando el promedio de sus tiempos de ejecución para realizar los gráficos siguientes. Cabe recalcar, que estas pruebas fueron hechas en computadores de 4 núcleos físicos y 4 núcleos virtuales, que se encuentran ubicados en los laboratorios del Departamento de Ingeniería Informática.

## 2.2. Gráficos

### 2.2.1. Hebras v/s Tiempo de Ejecución

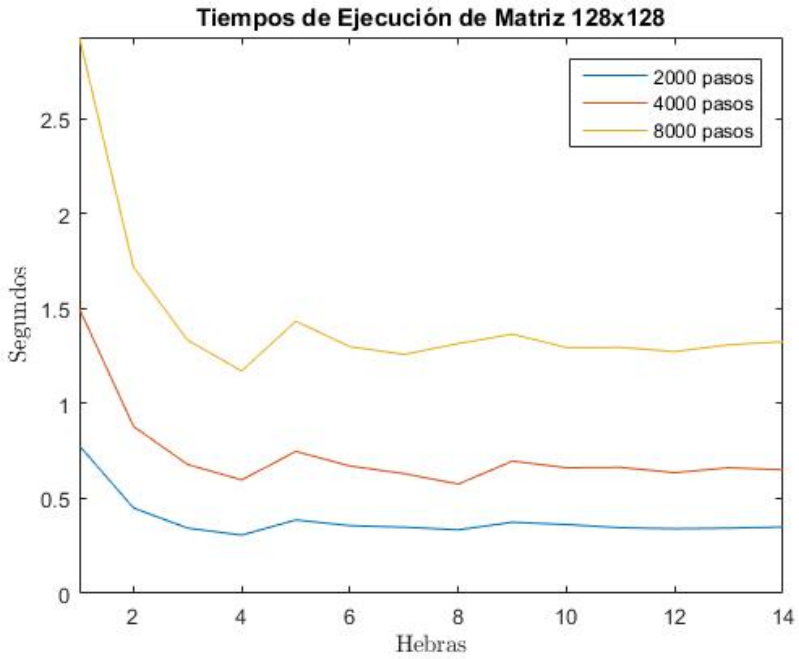


*Figura 2-1: Gráfico de Hebras v/s Tiempo de ejecución para la matriz de 128x128*

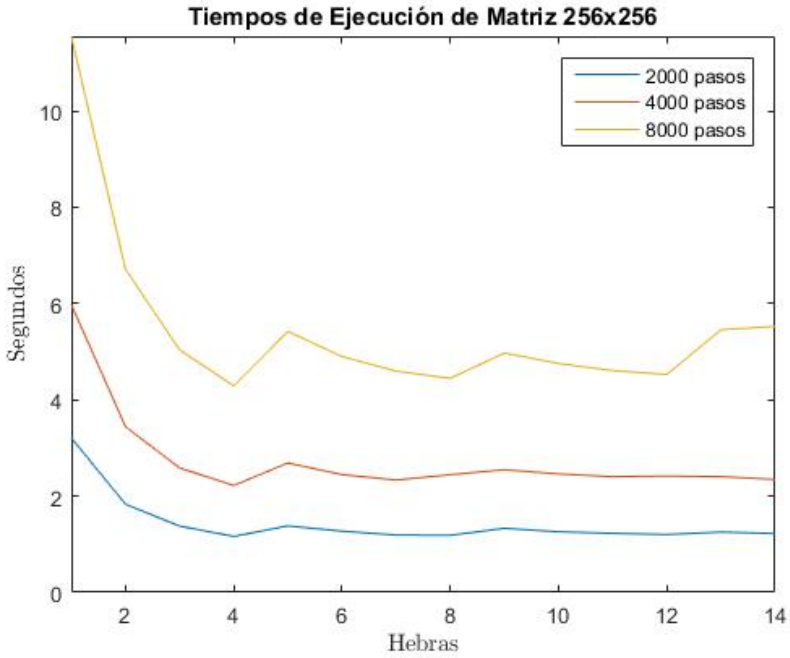


*Figura 2-2: Gráfico de Hebras v/s Tiempo de ejecución para la matriz de 256x256*

### 2.2.2. Hebras v/s Speed Up

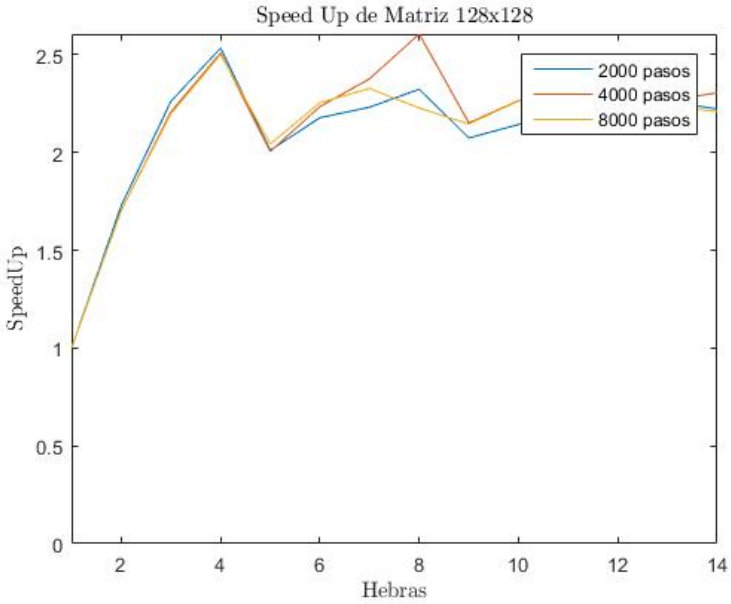


*Figura 2-3: Gráfico de Hebras v/s Speed Up para la matriz de 128x128*

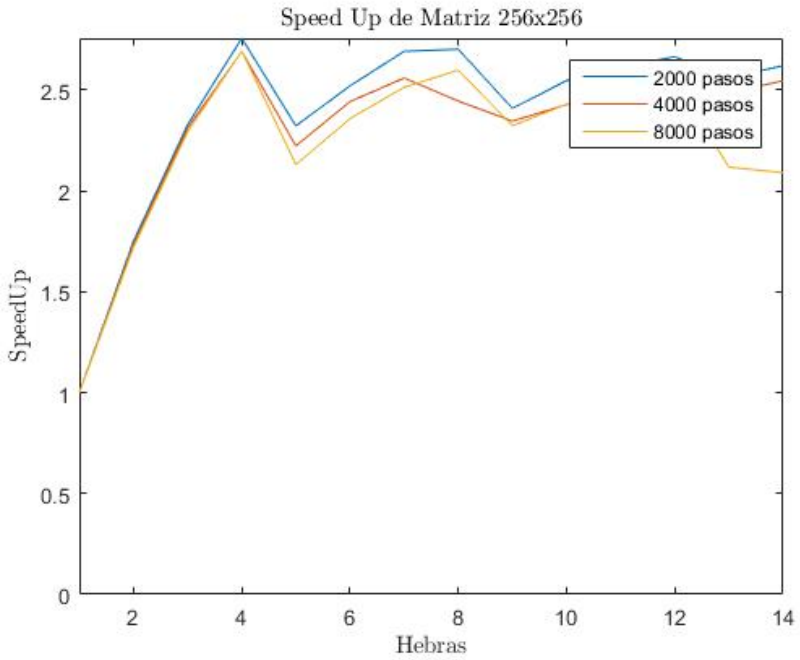


*Figura 2-4: Gráfico de Hebras v/s Speed Up para la matriz de 256x256*

### 2.2.3. Hebras v/s Eficiencia

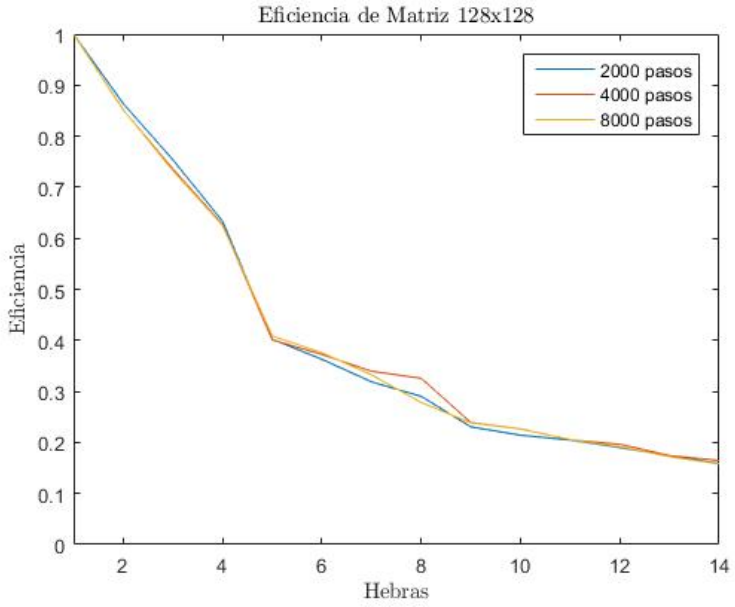


*Figura 2-5: Gráfico de Hebras v/s Eficiencia para la matriz de 128x128*

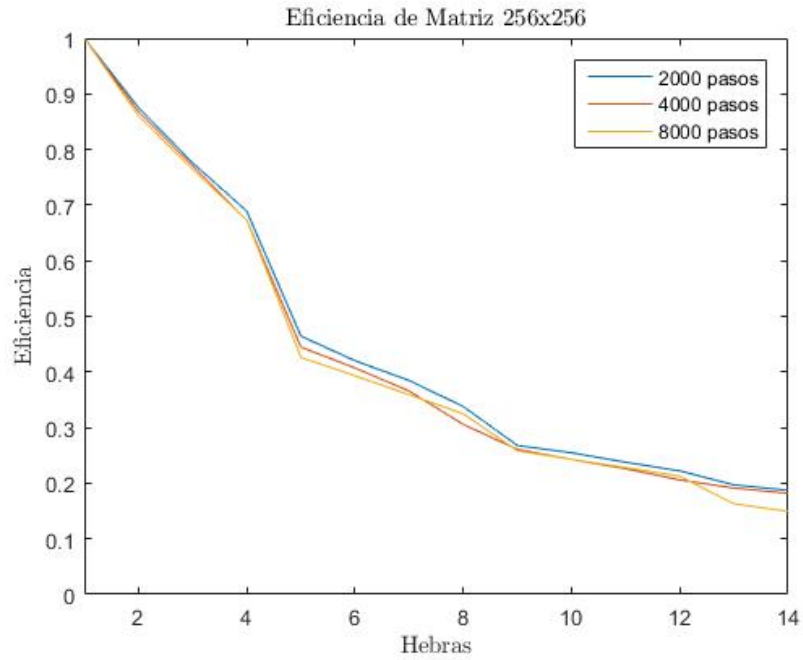


*Figura 2-6: Gráfico de Hebras v/s Eficiencia para la matriz de 256x256*

Apreciando los gráficos anteriores, se puede visualizar que la eficiencia para ambas matrices disminuye a medida que se agreguen cantidad de hebras, teniendo eficiencia máxima con 1 hebra. De esto se puede analizar que hasta 4 hebras se obtiene una eficiencia de sobre 0.5, es decir con el 50 % de eficiencia. Esto se encuentra en relación al Speed Up y al tiempo de ejecución, gráficos que también se encuentran anteriormente, donde el tiempo de ejecución para una hebra es la máxima en comparación a las demás hebras. A medida que se agregan hebras, el tiempo de ejecución disminuye hasta una cierta cantidad de hebras, donde el tiempo se mantiene aproximadamente constante, que en este caso para 4 hebras el tiempo de ejecución es la mejor, ya que son la cantidad de hebras que menor se demoran en terminar la ejecución del programa, para cualquier ancho y largo de matriz. Esto se debe a que el computador donde se hicieron las pruebas tienen 4 núcleos, y cada una de ellos ejecuta una sola hebra, por ende se pueden ejecutar 4 hebras paralelamente. Pero en caso de que la cantidad de hebras sea mayor o igual a 5, empiezan a existir cambios de contexto, lo cual aumenta poco el tiempo de ejecución haciendo que a medida que se agreguen hebras, sea menos eficiente.

Con el Speed Up se puede analizar que tanto mejora al agregar hebras y compararlo con un programa que es ejecutado solamente con una hebra. Para este caso, se interpreta de los gráficos que aumenta el Speed Up hasta que se agregan 4 hebras, luego tiende a oscilar dentro de los mismos valores, sin aumentar más allá de lo obtenido con estas 4.

En teoría, no debería aumentar más el Speed Up debido a que las hebras comienzan a interferir en el proceso del programa, es decir, se necesitan realizar más cambios de contexto mientras más hebras existan en el programa, ya que el máximo de hebras que se pueden ejecutar paralelamente son 4.

Cabe destacar, que los datos obtenidos son solamente una muestra y que podrían variar dependiendo de muchos factores, dentro de las cuales se encuentran la cantidad de núcleos del procesador, el planificador de Sistema Operativo, de la cantidad de programas en ejecución, etc.

Dentro de las mejoras que se puede realizar para este programa, se recalca de que se puede administrar de una mejor manera la memoria utilizada en el programa. Este cambio se vería reflejado en cambiar la estructura de datos ”wave.[en] la cual se crean ’n’ matrices, donde ’n’ corresponde a la cantidad de pasos o instancias donde se puede mejorar la ejecución utilizando solamente 3 matrices, las cuales se irán actualizando en cada iteración.